SYSTEM AND METHOD FOR DEFERRED REBALANCING OF A TREE DATA STRUCTURE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to co-pending application Ser. No. 09/729,515 entitled "Method and System for Enhanced Concurrency in a Computing Environment," by Clifford L. Hersh and Herbert W. Sullivan, filed on December 4, 2000, co-pending application Ser. No. 09/884,280 entitled "System and Method for Determining the Commutativity of Computational Operations," by Clifford L. Hersh, filed on June 19, 2001 and specifically incorporated herein by reference, co-pending application Ser. No. 09/884,243 entitled "System and Method for Managing Concurrent Operations on Linked Lists," by Clifford L. Hersh, filed on June 19, 2001, and co-pending application entitled "Multiphase System and Method of Performing Operations on Data Structures" filed on even date herewith.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not Applicable

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[0002] The invention relates generally to the field of computer algorithms and more particularly to a multiphase system and method of performing operations upon data structures.

BACKGROUND INFORMATION

[0003] Data structures, such as linked lists, trees, sets, and graphs are well known in the computing arts and are used to store information in a logical and efficient manner for access and operation by computing processes. Computing processes operable upon such data structures include operations for inserting, deleting, and updating data structure elements. Such computing processes include concurrent operations such as operations upon a data structure element and operations upon a row including the data structure element.

[0004] Systems and methods of the prior art avoid conflicts between concurrent operations by locking the affected data structure or a portion thereof affected by concurrent operations. In this manner the integrity of the data structure

being operated upon as well as the integrity of the information stored therein is ensured. Such locking further ensures that concurrent operations operating upon a same element of the data structure or upon the information stored in the same element are performed in the correct sequence to thereby eliminate corruption of the information or error conditions.

[0005] Thus for example, concurrent operations such as updating entries in a database are conventionally stored in an ordered list of pending operations. For each pending operation on the list, upon execution of the operation, an affected row in the database is locked until the operation is completed. The affected row is locked in order that other pending operations do not intercede and produce inconsistent results. For example, if the ordered list includes two updates to a cell in the database row, the affected row is locked while a first update operation executes and commits thus preventing a second update operation from concurrently executing. In this manner, the integrity of the database is maintained.

The execution of such operations using locks is expensive. Locking of the data structure can result in significant performance degradation of data structure operations because one operation may have to wait for another operation to complete. Additionally, while the data structure is locked, processes are precluded from reading information from the data structure.